Package ‘soundcorrs’

January 22, 2020

Title    Semi-Automatic Analysis of Sound Correspondences
Version  0.1.0
Description A set of tools that can be used in computer-aided analysis of sound correspondences between languages, plus several helper functions. Analytic functions range from purely qualitative analysis, through statistic methods yielding qualitative results, to an entirely quantitative approach.

Depends R (>= 3.5.0)
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Kamil Stachowski [aut, cre]
Maintainer Kamil Stachowski <kamil.stachowski@gmail.com>
Repository CRAN
Date/Publication 2020-01-22 15:20:02 UTC

R topics documented:

addSeparators .............................................. 2
allPairs ..................................................... 3
allTables .................................................... 4
binTable ..................................................... 5
cbind.soundcorrs ......................................... 5
char2value ................................................... 6
expandMeta .................................................. 7
findPairs ................................................... 7
findSegments ............................................... 8
### Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addSeparators</td>
<td>32</td>
</tr>
<tr>
<td>Intersperse a vector of strings with a character or string.</td>
<td>32</td>
</tr>
</tbody>
</table>

#### Description

Primarily intended to insert separators into a column of words, to facilitate manual segmentation and aligning.

#### Usage

```r
addSeparators(x, separator = "|")
```
allPairs

Arguments

x [character vector] The strings to be interspersed.
separator [character] The string with which to intersperse. Defaults to "|".

Value

character vector A vector of interspersed strings.

Examples

addSeparators (c("word","mot","focal"), ".")

allPairs Produce a list of all sound correspondences and all pairs in which they are attested.

Description

Take all segment-to-segment correspondences in the dataset, and produce for each a section composed of a title, a contingency table of all renderings of the given segment, and subsections listing all word pairs in which the given rendering is attested, all nicely formatted.

Usage

allPairs(data, file, count, unit, direction, cols, formatter, ...)

Arguments

data [character] The dataset. Only datasets with two languages are supported.
file [character] Name of the file to write the formatted list to. If NULL, the output will be printed to the screen. Defaults to NULL.
count [character] Report the absolute number of times or words, or relative to how many times or in how many words the given segments co-occur in L1 or L2. Accepted values are "a(bsolute)" and "r(elative)". Defaults to "a".
unit [character] Count how many times a correspondence occurs or in how many words it occurs. Accepted values are "o(ccurrence(s))" and "w(or(d(s)))". Defaults to "w".
direction [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.
cols [character vector] Which columns of the dataset to print. Can be a vector of names, "aligned" (the two columns with segmented, aligned words), or "all" (all columns). Defaults to "aligned".
formatter [function] The function to which to pass unformatted data. Available formatters are: formatter.none, formatter.html, and formatter.latex. Defaults to formatter.none.
...

Additional arguments passed to formatter.
allTables

Generate all contingency tables for a dataset.

Description

Generate all correspondence-to-correspondence or correspondence-to-metadata contingency tables for a dataset.

Usage

allTables(data, column, count, unit, direction, bin)

Arguments

data [soundcorrs] The dataset from which to draw frequencies. Only datasets with two languages are supported.

column [character] Name of the column with metadata. If NULL, sound correspondences are cross-tabulated with themselves. Defaults to NULL.

count [character] Report the absolute number of times or words, or relative to how many times or in how many words the given segments co-occur in L1 or L2. Accepted values are "a(bsolute)" and "r(elative)". Defaults to "a".

unit [character] Count how many times a correspondence occurs or in how many words it occurs. Accepted values are "o(ccurrence(s))" and "w(or(d(s)))". Defaults to "w".

direction [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.

bin [logical] Whether to bin tables before applying fun to them. Defaults to TRUE.

Value

list A list of tables.

Examples

dataset <- sampleSoundCorrsData.capitals
allPairs (dataset)
allPairs (dataset, formatter=formatter.latex, cols=c("ORTHOGRAPHY.German","ORTHOGRAPHY.Polish"))

dataset <- sampleSoundCorrsData.abc
allTables (dataset)
allTables (dataset, "DIALECT.L2", unit="o")
**binTable**

*Sum all rows and all columns in a table, except for the selected ones.*

**Description**

Useful for when the data are scarce and `chisq.test` returns a warning, or when a more specific analysis of the data is required.

**Usage**

```r
binTable(x, row, col)
```

**Arguments**

- `x` [data.frame/matrix/table] Table to be binned.
- `row` [integer/vector] The rows to not be binned.
- `col` [integer/vector] The columns to not be binned.

**Value**

table Table with some of its data binned.

**Examples**

```r
mtx <- matrix(1:16, nrow=4, dimnames=list(paste0("r",1:4),paste0("c",1:4)))
binTable(mtx, 1, 1)
binTable(mtx, 1, c(1,3))
```

---

**cbind.soundcorrs**

*Attach one or more columns to a soundcorrs object.*

**Description**

Attach one or more columns to a `soundcorrs` object. Note that sound correspondences attached with this function will not be usable as such.

**Usage**

```r
## S3 method for class 'soundcorrs'
cbind(data, ...)
```

**Arguments**

- `data` [soundcorrs] The `soundcorrs` object.
- `...` Objects to be attached.
Value

soundcorrs The original soundcorrs object with the columns attached.

Examples

dataset <- sampleSoundCorrsData.capitals
cbind (dataset, ID=1:nrow(dataset$data))
cbind (dataset, CONTINENT="Europe")

char2value

Convert characters to their values.

Description

Convert a vector of characters to their values, as defined in the transcription.

Usage

char2value(data, language, x)

Arguments

data [soundcorrs] The soundcorrs object which holds the transcription.
language [character or integer] Which transcription to use; can be the name of the language, or its number (in which place it was listed when creating the soundcorrs object.

x [character vector] Characters to convert.

Value

character vector Values, as defined in the transcription.

Examples

dataset <- sampleSoundCorrsData.abc
segms <- findSegments (dataset, "a", "o", +1)
char2value (dataset, "L1", segms$L1)
expandMeta

Expand custom metacharacters to a regular expression.

Description

Turn characters defined in a transcription as metacharacters into the corresponding regular expression.

Usage

\[
\text{expandMeta}(\text{transcription}, \text{x})
\]

Arguments

\[
\begin{align*}
\text{transcription} & \quad \text{[transcription]} \quad \text{The transcription to use.} \\
\text{x} & \quad \text{[character]} \quad \text{The string containing metacharacters.}
\end{align*}
\]

Value

character  A string with metacharacters expanded.

Examples

\[
\begin{align*}
\text{dataset} & \gets \text{sampleSoundCorrsData.capitals} \\
\text{expandMeta}(\text{dataset}\$\text{trans}[1], \text{"aN"}) \\
\text{orth.german} & \gets \text{dataset}\$\text{data}\$\text{ORTHOGRAPHY.German} \\
\text{query} & \gets \text{"lin"} \\
\text{orth.german} & [ \text{grep} (\text{query}, \text{orth.german}) ] \\
\text{query} & \gets \text{expandMeta}(\text{dataset}\$\text{trans}[1], \text{"lin$"}) \\
\text{orth.german} & [ \text{grep} (\text{query}, \text{orth.german}) ]
\end{align*}
\]

findPairs

Find all pairs with corresponding sequences of sounds.

Description

Sift the dataset for word pairs such that the first word contains x and the second word contains y in the corresponding segment or segments.

Usage

\[
\text{findPairs(data, x, y, exact, cols)}
\]
findSegments

Arguments

data [soundcors] The dataset in which to look. Only datasets with two languages are supported.
x [character] The sequence to find in language1. May be a regular expression. If an empty string, anything will be considered a match.
y [character] The sequence to find in language2. May be a regular expression. If an empty string, anything will be considered a match.
exact [logical] Only return exact, full-segment to full-segment matches? If TRUE, linguistic zeros are not ignored. Defaults to FALSE.
cols [character vector] Which columns of the dataset to return as the result. Can be a vector of names, "aligned" (the two columns with segmented, aligned words), or "all" (all columns). Defaults to "aligned".

Value
df.findPairs A subset of the dataset, containing only the pairs with corresponding sequences. Warning: pairs with multiple occurrences of such sequences are only included once.

Examples

# In the examples below, non-ASCII characters had to be escaped for technical reasons.
# In actual usage, all soundcors functions accept characters from beyond ASCII.
dataset <- sampleSoundCorrsData.capitals
findPairs (dataset, "\u00E4", "e", cols=c("ORTHOGRAPHY.German","ORTHOGRAPHY.Polish")) # a-diaeresis
findPairs (dataset, "a", "[ae]", cols="all")
findPairs (dataset, "\u0259", "Vr", exact=FALSE) # schwa
findPairs (dataset, "\u0259", "Vr", exact=TRUE) # schwa
subset (dataset, findPairs(dataset, "\u00E4", "e")$which) # a-diaeresis

findSegments Segments in relation to segments exhibiting a correspondence.

Description

Find pairs with a specific sound correspondence, and extract from them the segments which come before or after the segments exhibiting that correspondence.

Usage

findSegments(data, x, y, segment)
Arguments

data [soundcorrs] The dataset in which to look. Only datasets with two languages are supported.

x [character] The sequence to find in language 1. May be a regular expression. If an empty string, anything will be considered a match.

y [character] The sequence to find in language 2. May be a regular expression. If an empty string, anything will be considered a match.

segment [integer] Number of the segment to be returned, in relation to segments containing x and y. Defaults to 0.

Value

list Vectors for both languages, each of the same length as the dataset.

Examples

# In the examples below, non-ASCII characters had to be escaped for technical reasons.
# In actual usage, all soundcorrs functions accept characters from beyond ASCII.
dataset <- sampleSoundCorrsData.capitals
findPairs (dataset, "\u00E4", "e") # a-diaeresis
findSegments (dataset, "\u00E4", "e")
findSegments (dataset, "\u00E4", "e", -1)

fitTable Fit multiple models to multiple datasets.

Description

Apply multiFit to all rows or all columns of a table.

Usage

fitTable(models, data, margin, conv = vec2df.id, ...)

Arguments

models [list] A list of models to fit data to. Each element must be a list with at least two named fields: formula which contains the formula, and start which is a list of lists of starting estimates. Regarding the formula, the converter functions (fun, below) use "X" and "Y" for column names.

data [matrix/table] The data to fit models to.

margin [integer] As in apply: the subscripts which the fitting function (cf. multiFit) will be applied over. Accepted values are: 1 for rows, and 2 for columns.

conv [function] Function that converts vectors into data frames to which models will be fitted. Available functions are: vec2df.id, vec2df.hist, and vec2df.rank. Defaults to vec2df.id.

... Additional arguments passed to multiFit).
Value

list.multiFit A list of results returned by the fitting function (cf. multiFit).

Examples

dataset <- summary (sampleSoundCorrsData.abc)
models <- list (  
  "model A" = list (  
    formula = "Y ~ a/X",  
    start = list (list(a=1)))
  )
  "model B" = list (  
    formula = "Y ~ a/(1+exp(1)^X)"
  )
)
fitTable (models, dataset, 1, vec2df.rank)

formatter.html formatter.html

A formatter for allPairs. This one formats to HTML.

Description

A formatter for allPairs. This one formats to HTML.

Usage

formatter.html(what, x, direction = 1)

Arguments

what [character] What type of data is x.
x The object to be formatted.
direction [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.

Value

c character Formatted x.

Examples

dataset <- sampleSoundCorrsData.capitals
allPairs (dataset, unit="o", formatter=formatter.html)
**formatter.latex**

*A formatter for allPairs. This one formats to LaTeX.*

**Description**

A formatter for `allPairs`. This one formats to LaTeX.

**Usage**

`formatter.latex(what, x, direction = 1)`

**Arguments**

- `what` [character] What type of data is `x`.
- `x` The object to be formatted.
- `direction` [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.

**Value**

character Formatted x.

**Examples**

```r
dataset <- sampleSoundCorrsData.capitals
allPairs (dataset, unit="o", formatter=formatter.latex)
```

**formatter.none**

*A formatter for allPairs. This one does practically no formatting at all.*

**Description**

A formatter for `allPairs`. This one does practically no formatting at all.

**Usage**

`formatter.none(what, x, direction = 1)`

**Arguments**

- `what` [character] What type of data is `x`.
- `x` The object to be formatted.
- `direction` [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.
lapplyTest

Value
character Formatted x.

Examples

dataset <- sampleSoundCorrsData.capitals
allPairs (dataset, unit="o", formatter=formatter.none)

lapplyTest

Apply a function to a list.

Description
Takes a list and applies to each of its elements a function, returning a list of outputs. Primary intended for tests of independence on a list of contingency tables.

Usage

lapplyTest(x, fun = chisq.test, ...)

Arguments

x [list] The list to which to apply fun.
fun [function] The function which to apply to data. Must return an object containing an element named p.value. Defaults to chisq.test.
... Additional arguments passed to fun.

Value
list.lapplyTest A list of outputs of fun.

Examples

dataset <- sampleSoundCorrsData.abc
lapplyTest (allTables(dataset))
lapplyTest (allTables(dataset), fisher.test, simulate.p.value=TRUE)
long2wide

Convert from the long format (single entry per row) to the wide format (multiple entries per row).

Description

Takes a data frame of word pairs/triples/..., each stored in multiple rows, and returns a data frame with the same words but each pair/triple/... stored in one row. WARNING: in the original data frame, entries from all languages must be in the same order.

Usage

long2wide(data, col.lang = "LANGUAGE", skip = NULL)

Arguments

data [data.frame] The dataset to be converted.

col.lang [character] Name of the column with language names. Defaults to "LANGUAGE".

skip [character vector] Names of columns to not convert. Defaults to NULL.

Value
data.frame A data frame in the wide format (multiple entries per row).

Examples

# path to sample data in the "long format"
fName <- system.file("extdata", "data-abc.tsv", package="soundcorrs")
long <- read.table(fName, header=TRUE)
wide <- long2wide(long, skip=c("ID"))

multiFit

Fit multiple models to one dataset.

Description

Apply a fitting function, with multiple models and multiple starting estimates, to one dataset.

Usage

multiFit(models, data, fun = nls, ...)

Arguments

models [list] A list of models to fit data to. Each element must be a list with at least two named fields: formula which contains the formula, and start which is a list of lists of starting estimates.

data [numeric data.frame/list] A list of vectors to fit models to.

fun [function] The function to use for fitting. Defaults to nls.

... Additional arguments passed to fun.

Value

list.multiFit A list of results returned by fun or, if it ended with an error, NULL.

Examples

set.seed (27)
dataset <- data.frame (X=1:10, Y=(1:10)^2+runif(10,-10,10))
models <- list (  
  "model A" = list (  
    formula = "Y ~ X^a",  
    start = list (list(a=100), list(a=1)))  
  ,  
  "model B" = list (  
    formula = "Y ~ a*(X+b)",  
    start = list (list(a=1,b=1)))  
  )
multiFit (models, dataset)

ngrams

Frequencies of n-grams.

Description

Find n-grams of specified length and return their counts.

Usage

ngrams(data, n, zeros, as.table)

Arguments

data [scOne] A scOne object in which to look for n-grams.
n [integer] The length of n-grams to look for. Defaults to 1.
zeros [logical] Include linguistic zeros? Defaults to TRUE.
as.table [logical] Return the result as a table? Defaults to TRUE.

Value

table Table with counts of n-grams.
**print.df.findPairs**

Pretty printing for the result of `findPairs`.

### Description

Pretty printing for the result of `findPairs`.

### Usage

```r
## S3 method for class 'df.findPairs'
print(x, ...)
```

### Arguments

- `x` [df.findPairs] The output of `findPairs`.
- `...` Unused; only for consistency with `print`.

### Value

A more human-friendly digest.

### Examples

```r
dataset <- sampleSoundCorrsData.capitals
df.pair <- findPairs (dataset, "a", "[ae]", cols="all")
print(df.pair)
```

**print.scOne**

A more reasonable display of a `scOne` object.

### Description

A more reasonable display of a `scOne` object.

### Usage

```r
## S3 method for class 'scOne'
print(x, ...)
```
Arguments

x [scOne] The scOne object.
...

Value

A more human-friendly digest.

Examples

# path to sample data in the "wide format"
fNameData <- system.file("extdata", "data-capitals.tsv", package="soundcorrs")
# path to a sample transcription
fNameTrans <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
read.scOne (fNameData, "German", "ALIGNED.German", fNameTrans)
print.transcription  A more reasonable display of a transcription object.

Description

A more reasonable display of a transcription object.

Usage

```r
## S3 method for class 'transcription'
print(x, ...)
```

Arguments

- `x`  
  [transcription] The transcription.
- `...`  
  Unused; only for consistency with `print`.

Value

A more human-friendly digest.

Examples

```r
# path to a sample transcription
fName <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
read.transcription (fName)
```

---

read.scOne  Read data for a single language from a tsv file.

Description

Read the data for one language, from a file in the wide format, and combine it with metadata into a scOne object.

Usage

```r
read.scOne(file, name, col.aligned, transcription, separator = "\|")
```

Arguments

- `file`  
  [character] Path to the data file in the wide format.
- `name`  
  [character] Name of the language.
- `col.aligned`  
  [character] Name of the column with the aligned words.
- `transcription`  
  [character] Path to the file with the transcription.
- `separator`  
  [character] String used to separate segments in `col.aligned`. Defaults to "\|".
Value

scOne An object containing the data and metadata for one language.

Examples

# path to sample data in the "wide format"
fNameData <- system.file("extdata", "data-capitals.tsv", package="soundcorrs")
# path to a sample transcription
fNameTrans <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
ger <- read.scOne (fNameData, "German", "ALIGNED.German", fNameTrans)

read.transcription Read transcription from a tsv file.

Description

Read a table from file and create a transcription object out of it.

Usage

read.transcription(
  file,
  col.grapheme = "GRAPHEME",
  col.meta = "META",
  col.value = "VALUE"
)

Arguments

data, file [character] Path to the data file.
col.grapheme [character] Name of the column with graphemes. Defaults to "GRAPHEME".
col.meta [character] Name of the column with the coverage of metacharacters. If empty string or NA, the column will be generated automatically. Defaults to "META".
col.value [character] Name of the column with values of graphemes. Defaults to "VALUE".

Value

transcription A transcription object containing the read transcription.

Examples

# path to a sample transcription
fName <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
read.transcription (fName)
**sampleSoundCorrsData.abc**

*A sample dataset with entirely made up words and languages.*

---

**Description**

A sample dataset with entirely made up words and languages.

**Usage**

`sampleSoundCorrsData.abc`

**Format**

A `soundcorrs` object.

---

**sampleSoundCorrsData.capitals**

*A sample dataset with the names of EU capitals in German, Polish, and Spanish.*

---

**Description**

A sample dataset with the names of EU capitals in German, Polish, and Spanish.

**Usage**

`sampleSoundCorrsData.capitals`

**Format**

A `soundcorrs` object.
sampleSoundCorrsData.ie

*A sample dataset with a dozen words in English, Gothic, Greek, and Latin.*

---

**Description**

A sample dataset with a dozen words in English, Gothic, Greek, and Latin.

**Usage**

sampleSoundCorrsData.ie

**Format**

A `soundcorrs` object.

**Source**


---

**scOne**

Constructor function for the scOne class.

---

**Description**

Take a data frame containing data for one language, in the wide format, and combine it with metadata into a `scOne` object. In a normal workflow, the user should have no need to invoke this function other than through `read.scOne`.

**Usage**

```r
scOne(data, name, col.aligned, transcription, separator = "\|")
```

**Arguments**

- `data` [data.frame] Data for one language.
- `name` [character] Name of the language.
- `col.aligned` [character] Name of the column with the aligned words.
- `transcription` [transcription] The transcription for the given language.
- `separator` [character] String used to separate segments in `col.aligned`. Defaults to "\|".

**Value**

`scOne` A `scOne` object containing the data and metadata for one language.
Fields

cols [character list] Names of important columns.
data [data.frame] The original data.
names [character] Name of the language.
segms [character list] Words exploded into segments. With linguistic zeros preserved ($z$) or removed ($nz$).
segpos [integer list] A lookup list to check which character belongs to which segment. Counted with linguistic zeros preserved ($z$) and removed ($nz$).
separator [character] The string used as segment separator in col.aligned.
trans [transcription] The transcription object for the language.
words [character list] Words obtained by removing separators from the col.aligned column. With linguistic zeros ($z$) or without them ($nz$).

Examples

fNameData <- system.file("extdata", "data-capitals.tsv", package="soundcorrs")
fNameTrans <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
readData <- read.table(fNameData, header=TRUE)
readTrans <- read.transcription(fNameTrans)
ger <- scOne(readData, "German", "ALIGNED.German", readTrans)

soundcorrs

Constructor function for the soundcorrs class.

Description

Take multiple scOne objects and combine them into a single soundcorrs object.

Usage

soundcorrs(...)

Arguments

... [scOne] Multiple scOne objects to be combined.

Value

soundcorrs An object containing the data and metadata for multiple language.
Fields

cols [character list] Names of important columns.
data [data.frame] The original data, merged.
name [character] Names of the languages.
segms [character list] Words exploded into segments. With linguistic zeros preserved ($z$) or removed ($nz$).
segpos [integer list] A lookup list to check which character belongs to which segment. Counted with linguistic zeros preserved ($z$) and removed ($nz$).
separators [character] Strings used as segment separators in cols$aligned.
trans [transcription] transcription objects.
words [character list] Words obtained by removing separators from the cols$aligned columns. With linguistic zeros ($z$) or without them ($nz$).

Examples

# path to sample data in the "wide format"
fNameData <- system.file("extdata", "data-capitals.tsv", package="soundcorrs")
# path to a sample transcription
fNameTrans <- system.file("extdata", "trans-common.tsv", package="soundcorrs")
ger <- read.scOne (fNameData, "German", "ALIGNED.German", fNameTrans)
pol <- read.scOne (fNameData, "Polish", "ALIGNED.Polish", fNameTrans)
spa <- read.scOne (fNameData, "Spanish", "ALIGNED.Spanish", fNameTrans)
dataset <- soundcorrs (ger, pol, spa)

subset.soundcorrs

Return a subset of sound correspondences data which meets a condition.

Description

Reduce a soundcorrs object to just those word pairs/triples/... which meet a certain condition.

Usage

## S3 method for class 'soundcorrs'
subset(x, condition, ...)

Arguments

x [soundcorrs] The dataset to be subsetted.
condition [logical] The condition the subsetted data must meet.
... Unused; only for consistency with subset.

Value

soundcorrs A soundcorrs object containing the subsetted dataset.
A quick summary of the result of \texttt{lapplyTest}.

Description

 Take the output of \texttt{lapplyTest}, and extract from it only the noteworthy results.

Usage

\begin{verbatim}
## S3 method for class 'list.lapplyTest'
summary(object, p.value = 0.05, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{object} [list.lapplyTest] The output of \texttt{lapplyTest}.
  \item \texttt{p.value} [double] Results above this value will not be reported. Defaults to 0.05.
  \item ... Unused; only for consistency with \texttt{summary}.
\end{itemize}

Value

 A more human-friendly digest.

Examples

\begin{verbatim}
dataset <- sampleSoundCorrsData.abc
lapplyTest (allTables(dataset))
\end{verbatim}
A comparison of the results produced by `fitTable` or `multiFit`.

Description

Take the output of `fitTable` or `multiFit`, extract a specific metric from the fits, and present them in the form of a table.

Usage

```r
## S3 method for class 'list.multiFit'
summary(object, metric = "rss", ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>[list.multiFit] The output of <code>fitTable</code> or <code>multiFit</code>.</td>
</tr>
<tr>
<td>metric</td>
<td>[character] The metric to extract from <code>object</code>. Available metrics are: &quot;aic&quot;, &quot;bic&quot;, &quot;rss&quot;, and &quot;sigma&quot;. Defaults to &quot;rss&quot;.</td>
</tr>
<tr>
<td>...</td>
<td>Unused; only for consistency with <code>summary</code>.</td>
</tr>
</tbody>
</table>

Value

A more human-friendly digest.

Examples

```r
set.seed (27)
dataset <- data.frame (X=1:10, Y=(1:10)^2+runif(10,-10,10))
models <- list (
  "model A" = list (  
    formula = "Y ~ X*a",  
    start = list (list(a=100), list(a=1))),
  "model B" = list (  
    formula = "Y ~ a*(X+b)",  
    start = list (list(a=1,b=1)))
)
summary (multiFit(models,dataset))
summary (fitTable(models,as.matrix(dataset),1,vec2df.rank), "sigma")
```
**summary.soundcorrs**

*Generate a segment-to-segment contingency table for two languages.*

---

**Description**

Produce a contingency table detailing all segment-to-segment correspondences in a dataset.

**Usage**

```r
## S3 method for class 'soundcorrs'
summary(object, count = "a", unit = "w", direction = 1, ...)
```

**Arguments**

- `object` [soundcorrs] The dataset from which to draw frequencies. Only datasets with two languages are supported.
- `count` [character] Report the absolute number of times or words, or relative to how many times or in how many words the given segments co-occur in L1 or L2. Accepted values are "a(bsolute))" and "r(elative))". Defaults to "a".
- `unit` [character] Count how many times a correspondence occurs or in how many words it occurs. Accepted values are "o(ccurrence(s))" and "w(or(d)s))". Defaults to "w".
- `direction` [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x"). Defaults to 1.
- `...` Unused; only for consistency with `print`.

**Value**

`table` The contingency table.

**Examples**

```r
dataset <- sampleSoundCorrsData.abc
summary(dataset)
round(summary(dataset,count="r"), digits=3)
summary (dataset, unit="o")
summary (dataset, direction=2)
```
**Table**

*The base::table function.*

**Description**

The base::table function.

**Usage**

```r
table(...)
```

**Arguments**

...  

base::table's arguments.

---

**Table.soundcorrs**

*Generate a correspondence-to-correspondence or correspondence-to-metadata contingency table.*

**Description**

Take all segment-to-segment correspondences in a dataset, and cross-tabulate them with themselves or with metadata taken from a separate column.

**Usage**

```r
## S3 method for class 'soundcorrs'
table(data, column = NULL, count = "a", unit = "w", direction = 1, ...)
```

**Arguments**

- `data` [soundcorrs] The dataset from which to draw frequencies. Only datasets with two languages are supported.
- `column` [character] Name of the column with metadata. If NULL, sound correspondences are cross-tabulated with themselves. Defaults to NULL.
- `count` [character] Report the absolute number of times or words, or relative to how many times or in how many words the given segments co-occur in L1 or L2. Accepted values are "a(bsolute)" and "r(elative)". Defaults to "a".
- `unit` [character] Count how many times a correspondence occurs or in how many words it occurs. Accepted values are "o(rrurrence)" and "w(ords)". Defaults to "w".
- `direction` [integer] If 1, correspondences are in the order Language1 > Language2 ("x yields y"). If 2, the order is Language2 < Language1 ("y originates from x").
- `...` Unused; only for consistency with table.
transcription

Value

table  The contingency table.

Examples

dataset <- sampleSoundCorrsData.abc
table (dataset)
table (dataset, direction=2)
table (dataset, "DIALECT.L2")
round (table(dataset,"DIALECT.L2",count="r"), digits=3)

transcription  Constructor function for the transcription class.

Description

Take a data frame containing transcription and turn it into a transcription object, as required by the soundcorrs constructor function. In a normal workflow, the user should have no need to call this function other than through read.transcription.

Usage

transcription(
  data,
  col.grapheme = "GRAPHEME",
  col.meta = "META",
  col.value = "VALUE"
)

Arguments

data  [data.frame] Data frame containing the transcription and its meaning.
col.grapheme  [character] Name of the column with graphemes. Defaults to "GRAPHEME".
col.meta  [character] Name of the column with the coverage of metacharacters. If empty string or NA, the column will be generated automatically. Defaults to "META".
col.value  [character] Name of the column with values of graphemes. Defaults to "VALUE".

Value

transcription  A transcription object containing the provided data.

Fields

data  [data.frame] The original data frame.
cols  [character list] Names of the important columns in the data frame.
zero  [character] A regular expression to catch linguistic zeros.
vec2df.hist

A vector to data frame converter for fitTable. This one makes a histogram, and returns a data frame with midpoints and counts.

Description

A vector to data frame converter for fitTable. This one makes a histogram, and returns a data frame with midpoints and counts.

Usage

vec2df.hist(data)

Arguments

data [numeric vector] The data to be converted.

Value

data.frame Converted data.

Examples

dataset <- summary(sampleSoundCorrsData.abc)
models <- list (
  "model A" = list (
    formula = "Y ~ a/X",
    start = list (list(a=1))),
  "model B" = list (formula = "Y ~ a/(1 + exp(1)*X)",
    start = list (list(a=1)))
)
fitTable (models, dataset, 1, vec2df.hist)
vec2df.id  

A vector to data frame converter for `fitTable`. This one only does the necessary minimum.

**Description**

A vector to data frame converter for `fitTable`. This one only does the necessary minimum.

**Usage**

vec2df.id(data)

**Arguments**

- **data** [numeric vector] The data to be converted.

**Value**

data.frame  Converted data.

**Examples**

```r
dataset <- summary(sampleSoundCorrsData.abc)
models <- list(
  "model A" = list(
    formula = "Y ~ a/X",
    start = list(a=1)));
  "model B" = list(
    formula = "Y ~ a/(1+exp(1)\^X)",
    start = list(a=1))
)
fitTable(models, dataset, 1, vec2df.id)
```

vec2df.rank  

A vector to data frame converter for `fitTable`. This one orders data by rank.

**Description**

A vector to data frame converter for `fitTable`. This one orders data by rank.

**Usage**

vec2df.rank(data)
Arguments

data [numeric vector] The data to be converted.

Value
data.frame Converted data.

Examples

dataset <- summary(sampleSoundCorrsData.abc)
models <- list(
  "model A" = list(
    formula = "Y ~ a/X",
    start = list(list(a=1)))
  
  "model B" = list(
    formula = "Y ~ a/(1+exp(1)^X)",
    start = list(list(a=1)))
)
fitTable(models, dataset, 1, vec2df.rank)

wide2long

Convert from the wide format (multiple entries per row) to the long format (single entry per row).

Description

Takes a data frame of word pairs/triples/..., each stored in a single row, and returns a data frame with the same pairs/triples/... but with each word stored in its own row.

Usage

wide2long(data, suffixes, col.lang = "LANGUAGE", strip = 0)

Arguments

data [data.frame] The dataset to be converted.
suffixes [character vector] Suffixes used to differentiate column names; in the output, those will be used as language names.
col.lang [character] Name of the column in which language names are to be stored. Defaults to "LANGUAGE".
strip [integer] The number of characters to strip from the beginning of suffixes when they are turned into language names. Defaults to 0.

Value
data.frame A data frame in the long format (single entry per row).
Examples

# path to sample data in the "wide format"
fName <- system.file("extdata", "data-capitals.tsv", package="soundcorrs")
wide <- read.table(fName, header=TRUE)
long <- wide2long(wide, c(".German",".Polish",".Spanish"), strip=1)

%hasPrefix%  
Check if a string starts with another string.

Description

Within soundcorrs, primarily intended to extract rows and columns from contingency tables. Other than that, of general applicability.

Usage

x %hasPrefix% prefix

Arguments

x [character] The string or strings in which to look.
prefix [character] The string to look for. May be a regular expression.

Value

logical TRUE iff x begins with prefix.

Examples

"loans.tsv" %hasPrefix% "loans"
c("abc","bbc","cbc") %hasPrefix% "[bc]"

%hasSuffix%  
Check if a string ends in another string.

Description

Within soundcorrs, primarily intended to extract rows and columns from contingency tables. Other than that, of general applicability.

Usage

x %hasSuffix% suffix
Arguments

- **x** [character] The string or strings in which to look.
- **suffix** [character] The string to look for. May be a regular expression.

Value

logical TRUE iff x ends with suffix.

Examples

```r
"loans.tsv" %hasSuffix% ".tsv"
c("aba","abb","abc") %hasSuffix% "[bc]"
```
Index

*Topic dataset
  sampleSoundCorrsData.abc, 19
  sampleSoundCorrsData.capitals, 19
  sampleSoundCorrsData.ie, 20
  %hasPrefix%, 31
  %hasSuffix%, 31

  addSeparators, 2
  allPairs, 3, 10, 11
  allTables, 4
  apply, 9
  
  binTable, 5
  cbind.soundcorrs, 5
  char2value, 6
  chisq.test, 5, 12
  
  expandMeta, 7
  findPairs, 7, 15
  findSegments, 8
  fitTable, 9, 24, 28, 29
  formatter.html, 10
  formatter.latex, 11
  formatter.none, 11
  
  lapplyTest, 12, 23
  long2wide, 13
  
  multiFit, 9, 10, 13, 24
  
  ngrams, 14
  
  print, 15–17, 25
  print.df.findPairs, 15
  print.scOne, 15
  print.soundcorrs, 16
  print.transcription, 17
  
  read.scOne, 17, 20

  read.transcription, 18, 27
  sampleSoundCorrsData.abc, 19
  sampleSoundCorrsData.capitals, 19
  sampleSoundCorrsData.ie, 20
  scOne, 14–17, 20, 21
  soundcorrs, 5, 6, 16, 19, 20, 21, 22, 27
  subset, 22
  subset.soundcorrs, 22
  summary, 23, 24
  summary.list.lapplyTest, 23
  summary.list.multiFit, 24
  summary.soundcorrs, 25
  
  table, 26, 26
  table.soundcorrs, 26
  transcription, 6, 7, 17, 18, 20–22, 27
  
  vec2df.hist, 28
  vec2df.id, 29
  vec2df.rank, 29

  wide2long, 30